**Set 7**

The source code for the Critter class is in the critters directory

1. What methods are implemented in Critter?

**The act(), makeMove(), processActors(), selectMoveLocation(), getActors(), getMoveLocation().**

2. What are the five basic actions common to all critters when they act?

**Five basic common actions:  makeMove,  processActors, selectMoveLocation, getActors, getMoveLocation.**

3. Should subclasses of Critter override the getActors method? Explain.

**Yes, the subclasses of Critter have different behaviors, the different type Critter may get different**

**location's Actor, then it need to override the getActors() method.**

4. Describe the way that a critter could process actors.

**Different type of critter may have different processing ways. The default processing way is**

**If the actor is not a rock and not a critter, the critter can remove the actor from grid.**

**Another is change the color of the actor.**

5. What three methods must be invoked to make a critter move? Explain each of these methods.

**The three methods: getMoveLocation(), selectMoveLocation(), makeMove().**

**In the getMoveLocation,  it usethe getEmptyAdjacentLocations  to get a list of emptylocations**

**Then, in the selectMoveLocation(), it use a Math.random to selects a location to return. If no empty**

**location, it return the current location by using getLocation. Finally, in the makeMove() method, if the return location is no null, it move to the location.**

6. Why is there no Critter constructor?

**Since Critter is the subclass of Actor, Critter extends Actor, Actor has default constructor. Even there is
no constructor in Critter, when Critter is created, it would call the Actor's default constructor to create a
Critter.**

**Set 8**
The source code for the ChameleonCritter class is in the critters directory

1. Why does act cause a ChameleonCritter to act differently from a Critter even though

ChameleonCritter does not override act?

**Since ChameleonCritter override the processActors() and makeMove() method, and in the act method, it calls the processActors() and makeMove() and other methods. So the the act of ChameleonCritter is different from Critter.**

2. Why does the makeMove() method of ChameleonCritter call super.makeMove?

**Because in the makeMove() method of ChameleonCritter, it first change the the direction of the location, but not move the new location, only when it call the super.makeMove, it move the new location.**

3. How would you make the ChameleonCritter drop flowers in its old location when it moves?

**First, before moving the new location, obtain the old location.**

**Second, if the not valid location to move, do nothing, otherwise, after move the new location, then create a new flower, put it into the grid in the old location.**

```
If (!getLocation.equals(old_location))  {
        Flower flower = new Flower(getColor());
        flower.putSelfInGrid(grid, old_location);
}
```

4. Why doesn't ChameleonCritter override the getActors method?

**Because the ChameleonCritter does not define a new behaviors to get actors,  then it just inherits the getActor() from Critter to get list of actors.**

5. Which class contains the getLocation method?

**The Actor class contains the getLocation() method.**

6. How can a Critter access its own grid?

**A Critter access its own grid by calling the getGrid method which it inherits from the super class Actor.**

**Set 9**

The source code for the CrabCritter class is reproduced at the end of this part of GridWorld.

1. Why doesn't CrabCritter override the processActors method?

**Because the behaviors of CrabCritter processing to the actors returned by the getActors() is to eat them. That is same as the critter, since  CrabCritter have inherited from the critter , it does need to override the processActors() method.**

2. Describe the process a CrabCritter uses to find and eat other actors. Does it always eat all neighboring actors? Explain.

**It find the actors by calling the getActors(),  it  eats actors whatever is found in the locations immediately in front, to the right-front, or to the left-front of it by calling the processActors().**

**But it would not eat the rocksand the critters and neighbors in other location.**

3. Why is the getLocationsInDirections method used in CrabCritter?

**Because CrabCritter only eat the actors in the location in front, to the right-front , or to th left-front**

**of it, calling  getLocationsInDirections() method return the the those diretions to select the actors in those directions.**

4. If a CrabCritter has location (3, 4) and faces south, what are the possible locations for actors that are returned by a call to the getActors method?

**The actors in (4, 3), (4, 4), (4, 5)**

5. What are the similarities and differences between the movements of a CrabCritter and a Critter?

**Similarities: If the passing location loc is not the CrabCritter's current location, both of CrabCritter and Critter move to the a random location returned by the getActors(), and both of they not turns.**

**Differences: the Critter can move  to any direction but  the CrabCritter only moves to the front, right-front, left-front of it. When they both cant's move, the CrabCritter turns to right or left randomly**

**but the Critter does not turn.**

6. How does a CrabCritter determine when it turns instead of moving?

**If the passing loc in the makeMove() method  is not the CrabCritter's current location,**

**the crabCritter moves, otherwise it turns.**

7. Why don't the CrabCritter objects eat each other?

**Because the processActors() method of CrabCritter class is inherits from the Critter class, and in the processActor() of Critteer class,  it only eats the actors which is not a rock or a critters. Thus, the CrabCritter don't eat each other.**